**MEDICAL CHATBOT USING ML**

**A PROJECT REPORT**

***Submitted by***

|  |  |  |
| --- | --- | --- |
| **RITHIK K** | **IV CSE ‘B’** | **(610520104076)** |
| **SANJAY R** | **IV CSE ‘B’** | **(610520104077)** |
| **VIGNESH S** | **IV CSE ‘B’** | **(610520104103)** |
| **VIGNESH RAJ D** | **IV CSE ‘B’** | **(610520104104)** |

***In partial fulfilment for the award of the degree***

***of***

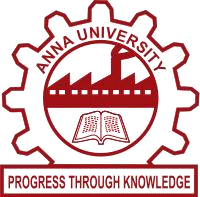
**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**DHIRAJLAL GANDHI COLLEGE OF TECHNOLOGY**

**SALEM - 636 309**



**ANNA UNIVERSITY :: CHENNAI 600 025**

**MAY 2024**

**MEDICAL CHATBOT USING ML**

**A FINAL YEAR PROJECT REPORT**

***In partial fulfilment for the award of the degree of***

***Submitted by***

|  |  |  |
| --- | --- | --- |
| **RITHIK K** | **IV CSE ‘B’** | **(610520104076)** |
| **SANJAY R** | **IV CSE ‘B’** | **(610520104077)** |
| **VIGNESH S** | **IV CSE ‘B’** | **(610520104103)** |
| **VIGNESH RAJ D** | **IV CSE ‘B’** | **(610520104104)** |

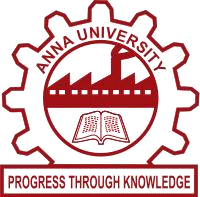
**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**DHIRAJLAL GANDHI COLLEGE OF TECHNOLOGY**

**SALEM - 636 309**



**ANNA UNIVERSITY:: CHENNAI 600 025**

**MAY 2024**

**BONAFIDE CERTIFICATE**

Certified that this project report **“MEDICAL CHATBOT USING ML”** is the bonafide work of **“ RITHIK K (610520104076), SANJAY R (610520104077), VIGNESH S (610520104103), VIGNESH RAJ D (610520104104)”** who carried out the work under my supervision.

|  |  |  |
| --- | --- | --- |
| **Dr.J.Vaijayanthimala, M.E.,Ph.D.,**  **HEAD OF THE DEPARTMENT**  Department of Computer Science and Engineering  Dhirajlal Gandhi College of Technology  Salem – 636 309. |  | **Ms.P. Vimala, M.E.,**  **SUPERVISOR**  Department of Computer Science and Engineering  Dhirajlal Gandhi College of Technology  Salem – 636 309. |

Submitted for University Viva Voce Examination held on ----------------

------------------------- -----------------------

**Internal Examiner External Examiner**

III

**ABSTRACT**

The idea is to create a medical chatbot that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. Our project focuses on providing the users immediate and accurate prediction of the diseases based on their symptoms. A multilingual chatbot is an AI- powered virtual assistant that can converse with customers in multiple languages .AI-powered chatbots can use advanced Natural Language Processing (NLP) algorithms to accurately interpret and deliver responses in any given foreign language. Some models feature pre-trained language models that allow them to mimic human conversation without needing extensive training. Voicebot are software powered by artificial intelligence (AI) that allow a caller to navigate an interactive voice response (IVR) system with their voice, generally using natural language.

IV

**திட்டப்பணிசுருக்கம்**

பிரதேசத்தில் ஒரு மருத்துவ சொல் போடுபவர் உட்கொண்டு நோய் அறிய மற்றும் மருத்துவ அறிக்கைகளைப் பராமரிக்கும் மருத்துவ சொல் போடுபவரை அழைக்கும் முன்னோடி மருத்துவ சொல் போடுபவர் உட்கொண்டு நோய் அறிக்கைகளைப் பராமரிக்க உதவும். இதன் மூலம் சுகாதாரத்தை குறைக்க மற்றும் மருத்துவ அறிவில் பாதுகாக்க அணுகலை மேம்படுத்த உதவும். எங்கள் திட்டம் பயனர்களுக்கு அவர்களின் அறிக்கைகளின் அடிப்படையில் உடைக்கப்படும் நோய்களின் உடைக்க தகுந்தளிக்க மற்றும் உள்ளீட்டு பரிசோதனைகளின் உடைக்க பயன்படுத்தும் மிகவும் அவசரமான மற்றும் உண்மையான கணிப்பு வழிகளை வழங்குகிறது. பல்வேறு மொழிகளில் பயனர்களுடன் உரையாடும் அறிக்கைகளை உடைக்க முடியும் தர்ம மருந்துவ சொல் போடுபவர் ஒரு AI-ஐயாக்கும் மாதிரி சொல் போடுபவர். ஏற்றுக்கொள்ளும் மொழி மயங்கி உரையாடும் அறிக்கைகளை சரியாக விளக்குகிறது. சில மாதிரிகள் முனைந்த மொழி மாதிரிகளை உள்ளிடும் முனைந்த மொழிகளில் தரவுத்த.

Top of Form

V

**ACKNOWLEDGEMENT**

Dedicating this project to the **ALMIGHTY GOD** whose abundant grace and mercy enabled the project’s successful completion.

We express our deep gratitude to our beloved **Shri. Dhirajlal A. Gandhi**, Founder Chairman, Dhirajlal Gandhi College of Technology, who gave us the golden opportunity to do this wonderful project.

We convey our sincere thanks to **Shri. D. Manoj Kumar**, Chairman, Dhirajlal Gandhi College of Technology, who supports us in all our endeavours.

We owe a genuine gratitude to **Smt. Archana Manoj Kumar**, **M.E**., Secretary, for providing all necessary facilities and guidance.

We are greatly indebted to thank to **Dr.A. Selvaraj, M.E., Ph.D., MISTE.,** principal, for providing support for this project.

We express our warm thanks to **Dr. J.Vaijayanthimala, M.E., Ph.D.,**

Head of the Department, Department of Computer Science and Engineering, for his encouragement and inspiration.

We are grateful to **Ms.P.Vimala, M.E.,** Assistant Professor, Department of Computer Science and Engineering, our supervisor, for her guidance and encouragement.

We are greatly thankful to **Mr.R.Makendran, M.E.,** Assistant Professor and **Mr.S.Sankar, M.E., MBA.,** Department of Computer Science and Engineering, our project coordinator, for their valuable support and guidance along the project.

VI

We are grateful to **Mr.S.A.AlthafAhamed,M.E.,** Assistant Professor, Department of Computer Science and Engineering, our class Advisors for their guidance and support.

Special thanks to our Faculty members, Supporting Staffs, Friends, our beloved Family and to those who are not mentioned above for the endless support given to us.

|  |  |  |
| --- | --- | --- |
|  |  | **RITHIK K**  **(610520104076)** |
|  |  | **SANJAY R**  **(610520104077)** |
|  |  | **VIGNESH S**  **(610520104103)** |
|  | VII | **VIGNESH RAJ D**  **(610520104104)** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| **1** | **INTRODUCTION**   * 1. Project detail     1.2 Project scope    1.3 Hardware and Software | **1**  **1**  **1** |
| **2** | **TECHNICAL FEASIBILITY**  2.1 Technical Feasibility  2.2 Technology description  2.2.1 HTML  2.2.2 CSS  2.2.3 JAVA SCRIPT  2.2.4 PYTHON  2.2.5 RASA | **3**  **4**  **4**  **5**  **5**  **7**  **9** |
| **3** | **ER DIAGRAM AND USECASE DIAGRAM**  3.1 ER diagram  3.2 Usecase diagram | **12**  **13** |
| **4** | **PLANNING AND IMPLEMENTATION**  4.1 Planning  4.2 Implementation | **17**  **20** |
|  | VIII |  |
| **5** | **FUTURE SCOPE AND CONCLUSION**  5.1 Future scope  5.2 conclusion | **24**  **24** |
|  | **APPENDIX – A** | **26** |
|  | **APPENDIX – B** | **58** |
|  | **REFERENCES** | **61** |

IX

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 3.1.1 | ER Diagram | 13 |
| 3.2.1 | Usecase Diagram-1 | 15 |
| 3.2.2 | Usecase Diagram-2 | 16 |
| Appendix-B-1 | Home Page | 58 |
| Appendix-B-2 | About Page | 58 |
| Appendix-B-3 | Contact Page | 59 |
| Appendix-B-4 | Chatbot Window | 59 |
| Appendix-B-5 | Chatbot output - 1 | 60 |
| Appendix-B-6 | Chatbot Output - 2 | 60 |
|  |  |  |

X

**LIST OF ABBREVIATIONS**

**HTML** Hyper Text Markup Language

**CSS** Cascading Style Sheets

**JS** Java Script

**RASA** Receive Appreciate Summarize Ask

**ER** Entity Relationship(Diagram)

**YAML** Yet Another Markup Language

XI

**CHAPTER 1**

**INTRODUCTION**

* 1. **Project detail**

The integration of artificial intelligence (AI) and machine learning (ML) techniques in healthcare has revolutionized patient care and medical services. This paper proposes the development and implementation of a Medical Chatbot empowered by ML algorithms to provide personalized and efficient healthcare support. The primary aim of this research is to enhance patient engagement, streamline communication between healthcare providers and patients, and offer reliable medical advice.

* 1. **Scope**

The scope of a medical chatbot using Rasa is broad and can be tailored to meet specific needs and objectives, whether for patient engagement, clinical support, or healthcare administration. Continuous refinement and iteration based on user feedback and technological advancements will be essential for ensuring the chatbot remains relevant and effective in addressing users' needs.

**1.3 Hardware Software Required**

To develop a medical chatbot using Rasa, you'll need both hardware and software components. Here's an overview of what you'll need:

**Hardware**:

**Development Machine**: You'll need a computer or server to develop and deploy your Rasa chatbot. The specifications of this machine will depend on the scale of your project and the number of users you expect to serve. Generally, a machine with sufficient RAM (at least 8GB), a multi-core CPU, and ample storage space should suffice for development purposes.

**Monitoring and Logging**:Implement monitoring tools to track server performance, resource utilization, and user interactions. Logging frameworks are essential for debugging, auditing, and compliance purposes.

**Software:**

•Operating System: Window 7 and above

• HTML, CSS, Java Script, (Front end)

• YAML (Database Tool)

• Python -Flask(Back end)

**CHAPTER 2**

**TECHNICAL FEASIBILITY**

* 1. **Technical feasibility**

Building a medical chatbot using Rasa is technically feasible and has been done successfully by many developers and organizations. Rasa is a powerful open-source framework for building conversational AI applications, offering robust features and flexibility for creating sophisticated chatbots tailored to specific use cases, including healthcare. Here's why building a medical chatbot using Rasa is technically feasible:

**Natural Language Understanding (NLU)**: Rasa's NLU capabilities enable the chatbot to understand user intents and extract relevant entities from user messages. This allows the chatbot to comprehend medical queries, symptoms, and requests for information or assistance.

**Dialogue Management**: Rasa's dialogue management capabilities enable the chatbot to maintain context, manage conversations, and generate appropriate responses based on the current dialogue state and user input. This is crucial for providing personalized assistance and guiding users through complex interactions, such as symptom checking or appointment scheduling.

**Customization and Flexibility**: Rasa provides a high degree of customization and flexibility, allowing developers to tailor the chatbot's behavior, responses, and workflows to suit specific healthcare scenarios and user needs. Custom actions, policies, and pipelines can be defined to implement specialized functionality and integrations with external systems.

**Integration Capabilities**: Rasa can integrate with various external systems and APIs, enabling the chatbot to access medical databases, appointment scheduling systems, electronic health records (EHR), telemedicine platforms, and other healthcare resources. This integration capability enhances the chatbot's utility and enables seamless interactions with existing healthcare infrastructure.

**2.2 Technology Description**

**2.2.1 HTML**

HTML or Hyper Text Markup Language is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

**2.2.2 CSS**

CSS was first developed in 1997, as a way for Web developers to define the look and feel of their Web pages. It was intended to allow developers to separate content from design so that HTML could perform more of the function that it was originally based on the markup of content, without worry about the design and layout.

CSS didn’t gain in popularity until around 2000, when Web browsers began using more than the basic font and color aspects of CSS.

Web Designers that don’t use CSS for their design and development of Web sites are rapidly becoming a thing of the past. And it is arguably as important to understand CSS as it is to know HTML – and some would say it was more important to know CSS.

Style sheet refers to the document itself. Style sheets have been used for document design for years. They are the technical specifications for a layout, whether print or online. Print designers use style sheets to insure that their designs are printed exactly to specifications. A style sheet for a Web page serves the same purpose, but with the added functionality of also telling the viewing engine (the Web browser) how to render the document being viewed.

**2.2.3 JAVASCRIPT**

JavaScript is a high-level, interpreted programming language primarily used for client-side web development. It allows developers to create dynamic and interactive web pages by adding behavior and functionality to HTML and CSS. JavaScript is one of the core technologies of the World Wide Web and is supported by all modern web browsers without requiring any additional plugins.

The full definition of JavaScript includes the following key aspects:

1. Syntax: JavaScript has a syntax similar to other programming languages such as C, C++, and Java. It uses a combination of statements, expressions, variables, functions, objects, and operators to define the logic and

behavior of a program.

2. Interpreted: JavaScript is an interpreted language, meaning that the code is executed directly without the need for explicit compilation. Web browsers interpret JavaScript code in real-time as it is encountered in the web page.

3. Client-Side Scripting: JavaScript primarily operates on the client side, allowing web developers to enhance the functionality and interactivity of web pages. It can manipulate the HTML Document Object Model (DOM) dynamically, handle user events, perform form validations, and make asynchronous requests to servers.

4. Object-Oriented: JavaScript is a prototype-based, object-oriented programming language. It provides built-in objects and supports the creation of custom objects using constructor functions or object literals. Objects encapsulate data and behavior, and inheritance allows for code reuse and modularity.

5. Functions: Functions in JavaScript are first-class citizens, meaning they can be assigned to variables, passed as arguments to other functions, and returned as values. JavaScript supports both named and anonymous functions, as well as higher-order functions, closures, and function expressions.

6. Browser Compatibility: JavaScript is supported by all major web browsers, including Chrome, Firefox, Safari, Edge, and Opera. However, different browsers may have slight variations in their JavaScript implementation, requiring developers to write code that is compatible across multiple browsers.

**2.2.4 PYTHON**

Python is a high-level programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code compared to languages like C++ or Java.

Key features of Python include:

**Readable and Concise Syntax**: Python's syntax is designed to be easily readable and concise, making it accessible to beginners and conducive to rapid development.

**Interpreted and Interactive**: Python is an interpreted language, meaning that code is executed line by line by the Python interpreter. This allows for interactive development and experimentation using tools like the Python shell or Jupyter notebooks.

**Dynamic Typing**: Python is dynamically typed, meaning that variable types are determined at runtime rather than being explicitly declared. This allows for more flexible and expressive code but may require careful attention to variable types to avoid unexpected behavior.

**Extensive Standard Library**: Python comes with a comprehensive standard library that provides modules and packages for a wide range of tasks, including file I/O, networking, data manipulation, web development, and more. This rich ecosystem of libraries accelerates development by providing pre-built solutions to common problems.

**Cross-platform Compatibility**: Python is available on multiple platforms, including Windows, macOS, and Linux, making it a versatile choice for developing applications that need to run on different operating systems.

**Object-Oriented Programming (OOP)**: Python supports object-oriented programming principles, allowing developers to create reusable and modular code through classes and objects.

**Functional Programming Features**: While primarily an object-oriented language, Python also incorporates functional programming features such as lambda functions, list comprehensions, and higher-order functions, enabling developers to write elegant and expressive code.

**Community and Ecosystem**: Python has a large and active community of developers who contribute to its ecosystem by creating libraries, frameworks, and tools for various domains, including web development, data science, machine learning, and scientific computing.

**2.2.5 RASA**

Rasa is an open-source framework for building conversational AI applications. It provides tools and libraries for developing chatbots, virtual assistants, and other natural language processing (NLP) applications. Rasa is designed to be flexible, scalable, and customizable, empowering developers to create sophisticated conversational experiences tailored to their specific use cases.

Key components of the Rasa framework include:

**Rasa Open Source**: The core component of the Rasa framework is Rasa Open Source, which includes libraries and tools for building conversational AI applications. This includes components for natural language understanding (NLU), dialogue management, and natural language generation (NLG).

**Rasa NLU**: Rasa NLU is a component of Rasa Open Source that provides tools for understanding user messages and extracting intents and entities. It uses machine learning techniques to classify user intents and extract relevant information from user inputs.

**Rasa Core**: Rasa Core is another component of Rasa Open Source that handles dialogue management. It allows developers to define conversational flows, handle context, and generate appropriate responses based on the current dialogue state and user input.

**Custom Actions**: Rasa allows developers to define custom actions, which are Python functions that can be executed as part of the conversation flow. Custom actions enable integration with external APIs, databases, and services, allowing chatbots to perform actions such as fetching data, making calculations, or interacting with external systems.

**Training Data**: Rasa uses training data in the form of conversation examples to train its NLU and dialogue management models. Developers provide annotated examples of user messages along with their corresponding intents and entities to teach the chatbot how to understand and respond to user inputs.

**Integration**: Rasa can be integrated with various messaging platforms, such as Slack, Facebook Messenger, or custom websites, allowing developers to

deploy their chatbots to different channels. Rasa also supports integration with other systems and services, enabling developers to build end-to-end conversational experiences.

**Extensions and Libraries**: Rasa provides a range of extensions, libraries, and tools to extend its functionality and simplify common tasks. This includes pre-built components for handling common NLP tasks, visualization tools for analyzing training data and model performance, and templates for building specific types of conversational AI applications.

**CHAPTER 3**

**ER AND DFD DIAGRAM**

* 1. **ER DIAGRAM**

Creating an Entity-Relationship (ER) diagram for a medical chatbot using Rasa involves identifying the entities and their relationships within the system. Here's a simplified ER diagram outlining the main entities and their relationships:

**User**:

Attributes: UserID (Primary Key), Username, Email, Age, Gender

Description: Represents individuals interacting with the chatbot, including patients and healthcare professionals.

**Symptom**:

Attributes: SymptomID (Primary Key), SymptomName, Description

Description: Represents various symptoms that users may report to the chatbot, such as cough, fever, headache, etc.

**Diagnosis**:

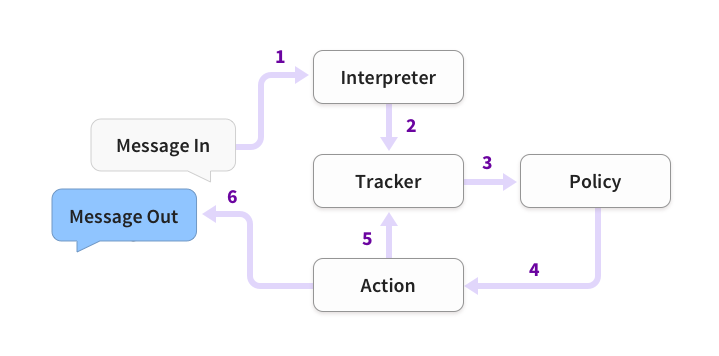
Attributes: DiagnosisID (Primary Key), DiagnosisName, Description

Description: Represents potential medical diagnoses suggested by the chatbot based on reported symptoms and other information provided by the user.

**Appointment**:

Attributes: AppointmentID (Primary Key), UserID (Foreign Key), Date, Time, Location, Purpose

Description: Represents appointments scheduled by users through the chatbot for medical consultations or other healthcare services.

****

**Fig.3.1.1 ER Diagram**

* 1. **Usecase Diagram**

Creating a use case diagram for a medical chatbot using Rasa can help illustrate the various interactions and functionalities of the system from the perspective of different actors. Here's a simplified use case diagram:

**Actors**:

**User**: Represents individuals interacting with the medical chatbot to seek information, schedule appointments, or receive healthcare assistance.

**Healthcare Provider**: Represents healthcare professionals who may interact with the chatbot to access patient information, provide guidance, or receive notifications.

**Use Cases**:

**Check Symptoms**: The user can input their symptoms, and the chatbot provides relevant information and guidance based on the reported symptoms.

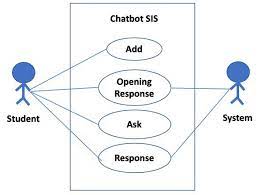
**Schedule Appointment**: The user can request to schedule a medical appointment, and the chatbot facilitates the appointment booking process by checking availability and confirming the appointment details.

**Medication Information**: The user can inquire about medications, dosages, and potential side effects, and the chatbot provides accurate and up-to-date information.

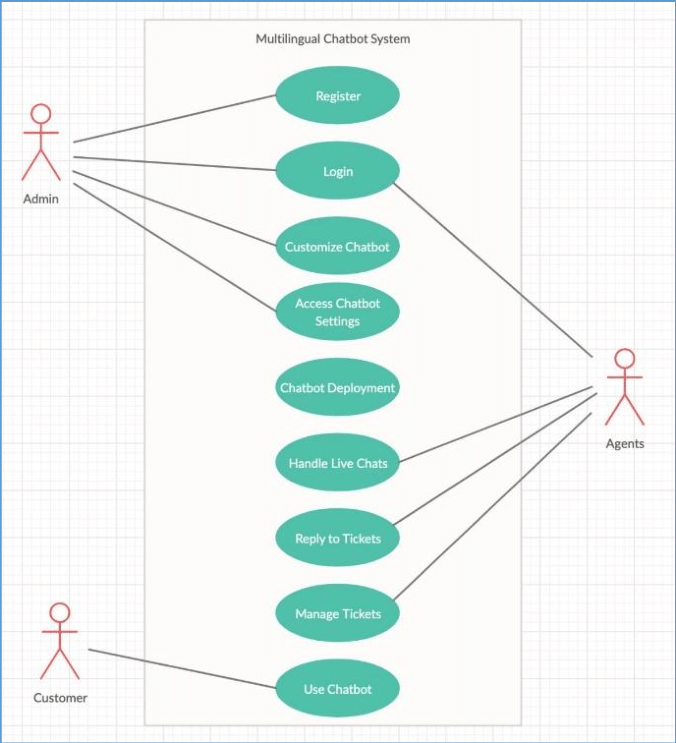
**Feedback Submission**: The user can provide feedback on their experience with the chatbot, including ratings and comments, to help improve the system.

**Access Medical Records**: The healthcare provider can access patient medical records and history through the chatbot, enabling them to provide informed care and recommendations.

**Receive Appointment Notifications**: The healthcare provider receives notifications about newly scheduled appointments or changes in patient status, allowing them to stay informed and prepared.

****

**Figure.3.2.1 Usecase Diagram-1**

****

**Figure .3.2.2 Usecase Diagram-2**

**CHAPTER 4**

**PLANNING AND IMPLEMENTATION**

* 1. **Planning**

Planning for a medical chatbot using Rasa involves several key steps to ensure a successful development process and deployment. Here's a structured approach to planning for a medical chatbot using Rasa:

**Define Objectives and Use Cases**:

Clearly define the objectives of the medical chatbot, such as providing symptom checking, appointment scheduling, medication reminders, or general health information.

Identify specific use cases that the chatbot will address, considering the needs of both users (patients) and healthcare professionals.

**Gather Requirements**:

Conduct research and gather requirements from stakeholders, including healthcare providers, patients, and other relevant parties.

Determine the features, functionalities, and integration points required for the chatbot to fulfill its objectives effectively.

**Design Conversational Flows**:

Design conversational flows for various use cases, outlining the sequence of interactions between the user and the chatbot.

Define intents, entities, and responses for each step of the conversation, considering different user inputs and possible scenarios.

**Data Collection and Annotation**:

Collect and annotate training data for the chatbot's natural language understanding (NLU) component, including examples of user messages, intents, and entities.

Annotate data for dialogue management, specifying the context and responses for different conversation paths.

**Setup Development Environment**:

Install Rasa Open Source and set up the development environment on your local machine or a cloud-based platform.

Create a new Rasa project and organize project files, including configuration files, training data, custom actions, and other resources.

**Develop NLU Model**:

Develop the natural language understanding (NLU) model using Rasa's NLU component.

Train the NLU model using annotated training data and evaluate its performance using metrics such as accuracy, precision, recall, and F1 score.

**Implement Dialogue Management**:

Implement dialogue management using Rasa's Core component, defining conversation flows and responses based on user inputs and dialogue context.

Train the dialogue management model using annotated dialogue data and fine-tune its behavior to ensure smooth and coherent conversations.

**Integrate External Systems**:

Integrate the chatbot with external systems and APIs, such as appointment scheduling systems, electronic health records (EHR), or medical databases.

Implement custom actions to handle interactions with external systems, retrieve relevant data, and perform necessary tasks.

**Testing and Validation**:

Conduct thorough testing of the chatbot's functionalities, including unit testing, integration testing, and end-to-end testing.

Validate the chatbot's performance with real users or domain experts, gathering feedback and making adjustments as needed.

**Deployment and Monitoring**:

Deploy the chatbot to production environments, such as messaging platforms, websites, or mobile apps.

Monitor the chatbot's performance, user interactions, and system metrics, and continuously improve its capabilities based on feedback and usage patterns.

* 1. **Implementation**

Implementing a medical chatbot using Rasa involves several steps, from setting up the development environment to deploying the chatbot to production. Here's a high-level overview of the implementation process:

**Setup Development Environment**:

Install Python and Rasa Open Source on your local machine or a cloud-based platform.

Create a new Rasa project using the Rasa Command Line Interface (CLI).

**Define Intents, Entities, and Responses**:

Define intents representing user intentions, such as "check symptoms," "schedule appointment," or "request medication information."

Identify entities representing important pieces of information in user messages, such as symptoms, dates, times, or locations.

Create responses for each intent, specifying the chatbot's replies or actions for different scenarios.

**Collect and Annotate Training Data**:

Collect training data in the form of example user messages, along with their corresponding intents and entities.

Annotate the training data using Rasa's training data format, specifying intents and entities for each example message.

**Develop Natural Language Understanding (NLU) Model**:

Configure the NLU pipeline in the Rasa configuration file, specifying components such as tokenizers, featurizers, and classifiers.

Train the NLU model using the annotated training data, using the Rasa CLI command **rasa train nlu**.

Evaluate the NLU model's performance using metrics such as accuracy, precision, recall, and F1 score.

**Implement Dialogue Management**:

Define the dialogue flow using stories, which represent sample conversations between the user and the chatbot.

Implement custom actions, which are Python functions that perform specific tasks or interact with external systems.

Train the dialogue management model using the annotated dialogue data and custom actions, using the Rasa CLI command **rasa train**.

**Integrate with External Systems**:

Implement custom actions to integrate the chatbot with external systems, such as appointment scheduling APIs, medical databases, or telemedicine platforms.

Handle authentication, data retrieval, and other interactions with external systems within custom action code.

**Testing and Validation**:

Conduct unit testing, integration testing, and end-to-end testing to validate the chatbot's functionalities and behavior.

Use Rasa's interactive learning mode to manually test and refine the chatbot's responses in real-time.

**Deployment to Production**:

Deploy the chatbot to production environments, such as messaging platforms (e.g., Slack, Facebook Messenger), websites, or mobile apps.

Configure webhooks or endpoints to receive and process user messages and deliver chatbot responses.

Monitor the chatbot's performance, user interactions, and system metrics in production to identify and address any issues or bottlenecks.

**Documentation and Maintenance**:

Document the chatbot's architecture, design decisions, implementation details, and deployment procedures for future reference and maintenance.

Establish procedures for ongoing maintenance, updates, and enhancements to ensure the chatbot remains effective and up-to-date over time.

**CHAPTER 5**

**FUTURE SCOPE & CONCLUSION**

* 1. **Future Scope**

The future scope for medical chatbots using Rasa is promising, with opportunities for further development and enhancements to improve healthcare accessibility, patient engagement, and clinical support. Here are some potential future directions and advancements for medical chatbots using Rasa:

**Advanced Natural Language Understanding (NLU)**:

Incorporate state-of-the-art NLU models and techniques, such as transformer-based architectures (e.g., BERT, GPT) or contextual embeddings (e.g., ELMo, Flair), to enhance the chatbot's understanding of complex medical queries and terminology.

**Personalized Health Recommendations**:

Implement personalized health recommendations and interventions based on individual user profiles, medical history, lifestyle factors, and real-time health data (e.g., wearable device data, electronic health records).

**5.2 Conclusion**

Medical chatbots using Rasa represent a valuable tool for transforming healthcare delivery, empowering individuals to take control of their health, and augmenting the capabilities of healthcare professionals to deliver patient-centered care. By embracing innovation, collaboration, and user-centric design principles, developers can harness the full potential of Rasa to create impactful and scalable solutions that address the evolving needs of patients and healthcare systems worldwide.

Leveraging Rasa's robust framework for developing medical chatbots holds immense potential to revolutionize healthcare accessibility and patient care delivery. By harnessing natural language understanding and dialogue management capabilities, Rasa-powered chatbots can efficiently triage patient inquiries, offer personalized health advice, schedule appointments, and provide vital information on symptoms and treatments. These chatbots not only enhance patient engagement but also streamline administrative processes for healthcare providers, driving efficiency and cost-effectiveness. Moreover, with continuous learning mechanisms, they can adapt and improve over time, ensuring relevance and effectiveness in dynamic healthcare environments. Ultimately, Rasa-based medical chatbots represent a transformative solution that empowers individuals to proactively manage their health while augmenting the capabilities of healthcare professionals to deliver high-quality care efficiently and effectively.

**APPENDIX – A**

**SOURCE CODE:**

**Domain.yml:**

version: "3.1"

intents:

- greet

- help

- symptom

- goodbye

- botchallenge

- fever

- bloodpressure

- diabetes

- cold

- heartsurgery

- eyepain

- earpain

- bodypain

- headache

- skinallergy

responses:

utter\_greet:

- text: "Hey! How are you?"

utter\_help:

- text: "okay."

utter\_help\_you:

- text: "How can i help you ?"

utter\_symptom:

- text: "What kind of Symptom you have ?"

utter\_goodbye:

- text: "Bye"

utter\_botchallenge:

- text: "I am a bot, powered by Rasa."

utter\_fever:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_fever\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_fever\_tablet:

- text: "Tablet name: DOLO 650"

image:"https://5.imimg.com/data5/SELLER/Default/2023/12/368132689/EB/NW/IA/83177447/dolo-650-mgeeee.jpg"

utter\_bloodpressure:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_bloodpressure\_tablet:

- text: "Tablet name: AMALONG A"

image: "https://www.netmeds.com/images/product-v1/600x600/331694/amlong\_a\_50mg\_tablet\_15s\_40422\_0\_1.jpg"

utter\_bloodpressure\_company:

- text: "Company name: Micro Labs Private Limited"

- text: "Okay.I will suggest you some tablets.Definitely that will cure you utter\_diabetes:

from your problem."

utter\_diabetes\_tablet:

- text: "Tablet name: TRIPRIDE 1"

image: "https://m.media-amazon.com/images/I/915gga1qRSL.jpg"

utter\_diabetes\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_cold:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_cold\_tablet:

- text: "Tablet name: NEW DOLO COLD"

image:"https://mahimaimedicals.myshopify.com/cdn/shop/products/New\_Dolo-cold\_Tab\_copy\_1024x1024@2x.jpg?v=1553149436"

utter\_cold\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_heartsurgery:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_heartsurgery\_tablet:

- text: "Tablet name: PLAGERINE A75"

image:https://onemg.gumlet.io/l\_watermark\_346,w\_480,h\_480/a\_ignore,w\_480,h\_480,c\_fit,q\_auto,f\_auto/ivqougae5sax6trlb4gm.jpg

utter\_heartsurgery\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_eyepain:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_eyepain\_eyedrop:

- text: "Eyedrop name: OFLACIN"

image:<https://images.apollo247.in/pub/media/catalog/product/o/f/ofl0051.jpg>

utter\_eyepain\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_bodypain:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_bodypain\_tablet:

- text: "Tablet name: DOLOWIN PLUS"

image: "https://m.media-amazon.com/images/I/81BnK4M7zVL.\_AC\_UF1000,1000\_QL80\_.jpg"

utter\_bodypain\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_earpain:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_earpain\_eardrop:

- text: "Eardrop name: OFLACIN"

image:https://images.apollo247.in/pub/media/catalog/product/o/f/ofl0051.jpg

utter\_earpain\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_headache:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_headache\_tablet:

- text: "Tablet name: DOLOPAR"

image: "https://www.netmeds.com/images/product-v1/600x600/312741/dolopar\_tablet\_15\_s\_0.jpg"

utter\_headache\_company:

- text: "Company name: Micro Labs Private Limited"

utter\_skinallergy:

- text: "Okay.I will suggest you some tablets.Definitely that will cure you from your problem."

utter\_skinallergy\_cream:

- text: "Cream name: MONO GUARD CREAM"

image: "https://www.practostatic.com/practopedia-images/v3/res-750/monoguard-cream-10gm\_547e9d3e-8c34-48d5-9057-9951263387ec.JPG"

utter\_skinallergy\_company:

- text: "Company name: Micro Labs Private Limited"

session\_config:

session\_expiration\_time: 60

carry\_over\_slots\_to\_new\_session: true

**nlu.yml:**

version: "3.1"

nlu:

- intent: greet

examples: |

- hey

- hello

- hi

- hello there

- good morning

- good evening

- hey there

- let's go

- hey dude

- goodmorning

- goodevening

- good afternoon

- intent: help

examples: |

- i am fine

- fine

- fyn

- intent: symptom

examples: |

- symptom

- having symptom

- i want to identify my problem

- problem

- symptom problem

- how to identify my problem

- how to identify my symptom

- intent: goodbye

examples: |

- cu

- good by

- cee you later

- good night

- bye

- goodbye

- have a nice day

- see you around

- bye bye

- see you later

- intent: botchallenge

examples: |

- are you a bot?

- are you a human?

- am I talking to a bot?

- am I talking to a human?

- intent: fever

examples: |

- fever

- suffering from fever

- not intrested in eating

- hot and dry skin

- vomiting

- intent: bloodpressure

examples: |

- blurred vision

- nose bleeds

- chest pain

- bp

- blood pressure

- intent: diabetes

examples: |

- diabetes

- are very hungry

- are very thirsty

- have blurry vision

- feel very tired

- intent: cold

examples: |

- cold

- suffering from cold

- blocked nose

- cough

- breathing difficulty

- intent: heartsurgery

examples: |

- heart surgery

- depression

- memory problems

- insomnia or difficulty sleeping

- muscle pain in chest area

- intent: eyepain

examples: |

- eyepain

- stabbing sensation

- eyeache

- throbbing

- aching

- intent: earpain

examples: |

- earpain

- trouble sleeping

- fussiness

- loss of balance

- earache

- intent: bodypain

examples: |

- bodypain

- joint aches and pains

- morning stiffness

- weakness

- bodyache

- intent: headache

examples: |

- headache

- head pain

- head usually hurts on both sides

- intent: skinallergy

examples: |

- allergy in skin

- rash

- itching

- redness

- swelling

**Stories.yml:**

version: "3.1"

stories:

- story: greet

steps:

- intent: greet

- action: utter\_greet

- story: help

steps:

- intent: help

- action: utter\_help

- action: utter\_help\_you

- story: symptom

steps:

- intent: symptom

- action: utter\_symptom

- story: goodbye

steps:

- intent: goodbye

- action: utter\_goodbye

- story: botchallenge

steps:

- intent: botchallenge

- action: utter\_botchallenge

- story: fever

steps:

- intent: fever

- action: utter\_fever

- action: utter\_fever\_tablet

- action: utter\_fever\_company

- story: bloodpressure

steps:

- intent: bloodpressure

- action: utter\_bloodpressure

- action: utter\_bloodpressure\_tablet

- action: utter\_bloodpressure\_company

- story: diabetes

steps:

- intent: diabetes

- action: utter\_diabetes

- action: utter\_diabetes\_tablet

- action: utter\_diabetes\_company

- story: cold

steps:

- intent: cold

- action: utter\_cold

- action: utter\_cold\_tablet

- action: utter\_cold\_company

- story: heartsurgery

steps:

- intent: heartsurgery

- action: utter\_heartsurgery

- action: utter\_heartsurgery\_tablet

- action: utter\_heartsurgery\_company

- story: eyepain

steps:

- intent: eyepain

- action: utter\_eyepain

- action: utter\_eyepain\_eyedrop

- action: utter\_eyepain\_company

- story: earpain

steps:

- intent: earpain

- action: utter\_earpain

- action: utter\_earpain\_eardrop

- action: utter\_earpain\_company

- story: bodypain

steps:

- intent: bodypain

- action: utter\_bodypain

- action: utter\_bodypain\_tablet

- action: utter\_bodypain\_company

- story: headache

steps:

- intent: headache

- action: utter\_headache

- action: utter\_headache\_tablet

- action: utter\_headache\_company

- story: skinallergy

steps:

- intent: skinallergy

- action: utter\_skinallergy

- action: utter\_skinallergy\_cream

- action: utter\_skinallergy\_company

**Rules.yml:**

version: "3.1"

rules:

- rule: Say goodbye anytime the user says goodbye

steps:

- intent: goodbye

- action: utter\_goodbye

- rule: Say 'I am a bot' anytime the user challenges

steps:

- intent: botchallenge

- action: utter\_botchallenge

**Index.html:**

<html>

<head>

<title>Medical Bot</title>

<!--Let browser know website is optimized for mobile-->

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<!--use for datepicker-->

<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>

<link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet">

<!--Import Google Icon Font-->

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

<link href="https://fonts.googleapis.com/css?family=Raleway:500&display=swap" rel="stylesheet">

<!--Import Font Awesome Icon Font-->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" integrity="sha256-eZrrJcwDc/3uDhsdt61sL2oOBY362qM3lon1gyExkL0=" crossorigin="anonymous" />

<!--Import materialize.css-->

<link rel="stylesheet" type="text/css" href="static/css/materialize.min.css">

<!--Main css-->

<link rel="stylesheet" type="text/css" href="static/css/style.css">

<meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body style="background-color: aqua;">

<div style="line-height: 75px; background-color: black; font-size: 20px; gap: 10px; justify-content: space-between;">

<a style="color: yellow;" href="">Home</a>

<a style="color: yellow; margin-left: 50px;" href="">About</a>

<a style="margin-left: 50px; margin-right: 50px; color: yellow;" href="">Contact</a>

</div>

<h1 style="color: olivedrab; text-align: center; font-size: 30px;">HOME</h1>

<h1 style="color: orange; font-size: 50px;">"Answers, care, and support on demand: Your medical chatbot friend."

</h1>

<p style="color: orangered; font-size: 30px;">A medical chatbot is a computer program designed to interact with users in natural language, providing information and assistance related to healthcare. It can offer guidance on symptoms, suggest potential diagnoses, provide treatment recommendations, schedule appointments, and offer general health advice. In the short term, these chatbots aim to improve access to healthcare information, offer preliminary assessments, and streamline communication between patients and healthcare providers. They rely on artificial intelligence and natural language processing techniques to understand and respond to user queries effectively.

</p>

<br>

<h1 style="color: blue; text-align: center; font-size: 30px;">ABOUT</h1>

<p style="color: blue; font-size: 30px;">A medical chatbot utilizing machine learning (ML) is a software application designed to interact with users in a conversational manner, providing information, guidance, or assistance related to medical queries or concerns. NLP is a core component of the chatbot that allows it to understand and interpret user input. It involves techniques like tokenization, entity recognition, and sentiment analysis to comprehend the user's message accurately.Given the sensitive nature of medical information, robust privacy and security measures must be in place to ensure the confidentiality and integrity of user data. Compliance with regulations such as HIPAA (in the United States) is crucial.

</p>

<h1 style="color: red; text-align: center; font-size: 30px;">CONTACT</h1>

<br><form>

<label for="fname">First Name</label>

<input type="text" id="fname" name="firstname" placeholder="Your name.." required>

<label for="lname">Last Name</label>

<input type="text" id="lname" name="lastname" placeholder="Your last name.." required>

<label for="country">Country</label>

<select id="country" name="country">

<option value="India">India</option>

<option value="canada">Canada</option>

<option value="usa">USA</option>

</select>

<label for="subject">Subject</label>

<textarea id="subject" name="subject" placeholder="Write something.." style="height:200px" required></textarea>

<input type="submit" value="Submit">

</form>

<div class="container">

<!-- Modal for rendering the charts, declare this if you want to render charts, else you remove the modal -->

<div id="modal1" class="modal">

<canvas id="modal-chart"></canvas>

</div>

<!--chatbot widget -->

<div class="widget">

<div class="chat\_header">

<!--Add the name of the bot here -->

<span class="chat\_header\_title">MEDICAL CHATBOT</span>

<span class="dropdown-trigger" href='#' data-target='dropdown1'>

<i class="material-icons">

more\_vert

</i>

</span>

<!-- Dropdown menu-->

<ul id='dropdown1' class='dropdown-content'>

<li><a href="#" id="clear">Clear</a></li>

<li><a href="#" id="restart">Restart</a></li>

<li><a href="#" id="close">Close</a></li>

</ul>

</div>

<!--Chatbot contents goes here -->

<div class="chats" id="chats">

<div class="clearfix"></div>

</div>

<!--keypad for user to type the message -->

<div class="keypad">

<textarea id="userInput" placeholder="Type a message..." class="usrInput"></textarea>

<div id="sendButton"><i class="fa fa-paper-plane" aria-hidden="true"></i></div>

</div>

</div>

<!--bot profile-->

<div class="profile\_div" id="profile\_div">

<img class="imgProfile" src="C:\Users\ADMIN\Medical Chatbot\RASAChatbot-main\RASAChatbot-main\ChatbotCode\rasa chatbot widget\New folder\img\chatbot.png" />

</div>

<!-- Bot pop-up intro -->

<div class="tap-target" data-target="profile\_div">

<div class="tap-target-content">

<h5 class="white-text">Hey there 👋</h5>

<p class="white-text">I can help you get started with Rasa and

answer your questions.</p>

</div>

</div>

</div>

<!--JavaScript at end of body for optimized loading-->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script type="text/javascript" src="static/js/materialize.min.js"></script>

<!--Main Script -->

<script type="text/javascript" src="static/js/script.js"></script>

<!--Chart.js Script -->

<script type="text/javascript" src="static/js/chart.min.js"></script>

</body>

</html>

**Style.css:**

body {

font-family: 'Raleway', sans-serif;

background: url('https://beingdatum.com/') no-repeat center center fixed;

background-size: cover;

}

/\* ============= css related to chats =============== \*/

.eshimg{

width:25%;

}

.widget {

display: none;

width: 350px;

right: 15px;

height: 500px;

bottom: 5%;

position: fixed;

background: #f7f7f7;

border-radius: 10px 10px 10px 10px;

box-shadow: 0 0px 1px 0 rgba(0, 0, 0, 0.16), 0 0px 10px 0 #00000096;

}

.chat\_header {

height: 65px;

background: rgb(90, 23, 238);

border-radius: 10px 10px 0px 0px;

padding: 5px;

font-size: 20px;

}

.chat\_header\_img{

margin-bottom: 15px;

}

.imgs{

width:75px;

}

.chat\_header\_title {

color: white;

margin-left: 45px;

}

.chats {

/\* display: none; \*/

height: 385px;

padding: 2px;

border-radius: 1px;

overflow-y: scroll;

margin-top: 1px;

transition: 0.2s;

}

div.chats::-webkit-scrollbar {

overflow-y: hidden;

width: 0px;

/\* remove scrollbar space /

background: transparent;

/ optional: just make scrollbar invisible \*/

}

.clearfix {

margin-top: 2px;

margin-bottom: 2px;

}

.botAvatar {

border-radius: 50%;

width: 1.5em;

height: 1.5em;

float: left;

margin-left: 5px;

/\* border: 2px solid #2c53af ; \*/

}

.botMsg {

float: left;

margin-top: 5px;

background: white;

color: black;

box-shadow: 2px 3px 9px 0px #9a82842e;

margin-left: 0.5em;

padding: 10px;

border-radius: 1.5em;

max-width: 60%;

min-width: 25%;

font-size: 13px;

word-wrap: break-word;

border-radius: 0 20px 20px 20px;

}

textarea {

box-shadow: none;

resize: none;

outline: none;

overflow: hidden;

font-family: Raleway;

}

textarea::-webkit-input-placeholder {

font-family: Raleway;

}

textarea-webkit-scrollbar {

width: 0 !important

}

.userMsg {

animation: animateElement linear 0.2s;

animation-iteration-count: 1;

margin-top: 5px;

word-wrap: break-word;

padding: 10px;

float: right;

margin-right: 0.5em;

background: #5a17ee;

color: white;

margin-bottom: 0.15em;

font-size: 13px;

max-width: 65%;

min-width: 15%;

border-radius: 20px 0px 20px 20px;

box-shadow: 0px 2px 5px 0px #9a828454;

}

.userAvatar {

animation: animateElement linear 0.3s;

animation-iteration-count: 1;

border-radius: 50%;

width: 1.5em;

height: 1.5em;

float: right;

margin-right: 5px;

/\* border: 2px solid #2c53af ; \*/

}

.usrInput {

padding: 0.5em;

width: 80%;

margin-left: 4%;

border: 0;

padding-left: 15px;

height: 40px;

}

.keypad {

/\* display: none; \*/

background: white;

height: 50px;

position: absolute;

bottom: 0px;

width: 100%;

padding: 5px;

border-radius: 0px 0px 10px 10px;

}

#sendButton {

height: 20px;

width: 20px;

border-radius: 50%;

display: inline-block;

float: right;

margin-right: 30px;

text-align: center;

padding: 5px;

font-size: 20px;

cursor: pointer;

color: #2b2bab;

}

.imgProfile {

box-shadow: 0 0px 1px 0 rgba(0, 0, 0, 0.16), 0 0px 10px 0 #00000096;

border-radius: 50px;

width: 60%;

}

.profile\_div {

/\* display: none; \*/

position: fixed;

padding: 5px;

width: 10%;

bottom: 0;

right: 0;

cursor: pointer;

}

/\* css for dropdown menu of the bot \*/

#close,

#restart,

#clear {

cursor: pointer;

}

.dropdown-trigger {

cursor: pointer;

color: white;

margin-right: 5px;

float: right;

margin-top: 3%;

}

.dropdown-content li>a,

.dropdown-content li>span {

color: #5a17ee;

}

@keyframes animateElement {

0% {

opacity: 0;

transform: translate(0px, 10px);

}

100% {

opacity: 1;

transform: translate(0px, 0px);

}

}

/\* css for card details of carousel cards \*/

.content {

transition: 0.2s;

display: none;

width: 350px;

right: 15px;

height: 500px;

bottom: 5%;

position: fixed;

background: white;

border-radius: 10px 10px 10px 10px;

box-shadow: 0px 2px 10px 1px #b5b5b5;

}

.content\_header {

object-fit: cover;

width: 100%;

height: 40%;

border-radius: 10px 10px 0px 0px;

}

.content\_data {

overflow-y: scroll;

height: 70%;

position: absolute;

left: 0px;

width: 100%;

top: 30%;

z-index: 1000;

border-radius: 10px;

background: white;

padding: 5px;

box-shadow: 0px -1px 20px 3px #9c9393ba;

}

.content\_title {

color: black;

font-weight: 600;

word-wrap: break-word;

padding-left: 5px;

font-size: 1.2em;

width: 80%;

border-radius: .28571429rem;

}

.votes {

font-size: 12px;

color: lightslategray;

}

.ratings {

margin-top: 5px;

background: #9ACD32;

padding: 5px;

color: white;

border-radius: 5px;

}

.user\_ratings {

border-radius: .28571429rem;

color: #fff;

font-weight: 600;

font-size: 15px;

}

.total\_ratings {

font-size: 12px;

opacity: .5;

margin-left: 5px;

}

.content\_data>.row .col {

padding: 5px;

}

.metadata\_1,

.metadata\_2,

.metadata\_3,

.metadata\_4,

.row {

margin-bottom: 0px;

}

.metadata\_1 {

color: lightslategrey;

padding: 5px;

}

.order,

#closeContents {

color: #2c53af;

}

.metadata\_2,

.metadata\_3,

.metadata\_4,

.metadata\_5,

.metadata\_6 {

color: lightslategrey;

padding: 5px;

}

.average\_cost,

.timings,

.location,

.cuisines {

width: 70%;

float: right;

margin-right: 25%;

}

.fa .fa-user-o {

font-size: 15px;

}

.stars-outer {

display: inline-block;

position: relative;

font-family: FontAwesome;

}

.stars-outer::before {

content: "\f006 \f006 \f006 \f006 \f006";

}

.stars-inner {

position: absolute;

top: 0;

left: 0;

white-space: nowrap;

overflow: hidden;

width: 0;

}

.stars-inner::before {

content: "\f005 \f005 \f005 \f005 \f005";

color: #f8ce0b;

}

div.content::-webkit-scrollbar {

width: 0 !important

}

div.content\_data::-webkit-scrollbar {

overflow-y: hidden;

width: 0px;

/\* remove scrollbar space /

background: transparent;

/ optional: just make scrollbar invisible \*/

}

/\* css for single card \*/

.singleCard {

padding-left: 10%;

padding-right: 10px;

}

/\* css for image card \*/

.imgcard {

object-fit: cover;

width: 80%;

height: 50%;

border-radius: 10px;

margin-left: 1%;

}

/\* css for suggestions buttons \*/

.suggestions {

padding: 5px;

width: 80%;

border-radius: 10px;

background: #ffffff;

box-shadow: 2px 5px 5px 1px #dbdade;

}

.menuTitle {

padding: 5px;

margin-top: 5px;

margin-bottom: 5px;

}

.menu {

padding: 5px;

}

.menuChips {

display: block;

background: #2c53af;

color: #fff;

text-align: center;

padding: 5px;

margin-bottom: 5px;

cursor: pointer;

border-radius: 15px;

font-size: 14px;

word-wrap: break-word;

}

/\* cards carousels \*/

.cards {

display: none;

position: relative;

max-width: 350px;

}

.cards\_scroller {

overflow-x: scroll;

overflow-y: hidden;

display: flex;

height: 260px;

width: 360px;

transition: width 0.5s ease;

margin-left: 5px;

/\* Enable Safari touch scrolling physics which is needed for scroll snap \*/

-webkit-overflow-scrolling: touch;

}

.cards\_scroller img {

border-radius: 10px;

}

.cards div.note {

position: absolute;

/\* vertically align center \*/

top: 50%;

transform: translateY(-50%);

left: 0;

right: 0;

background: rgba(0, 0, 0, 0.6);

padding: 20px;

position: absolute;

font-size: 4em;

color: white;

}

.cards .arrow {

position: absolute;

top: 50%;

transform: translateY(-50%);

height: 30px;

width: 30px;

border-radius: 2px;

background-position: 50% 50%;

background-repeat: no-repeat;

z-index: 1;

}

.cards .arrow.next {

display: none;

font-size: 2em;

color: #ffffff;

right: 10px;

}

.cards .arrow.prev {

display: none;

font-size: 2em;

color: #ffffff;

left: 10px;

}

.cards\_scroller::-webkit-scrollbar {

width: 0 !important

}

.cards\_scroller>div.carousel\_cards {

margin: 5px;

scroll-snap-align: center;

position: relative;

scroll-behavior: smooth;

}

.cards\_scroller div.carousel\_cards {

min-width: 50%;

min-height: 70%;

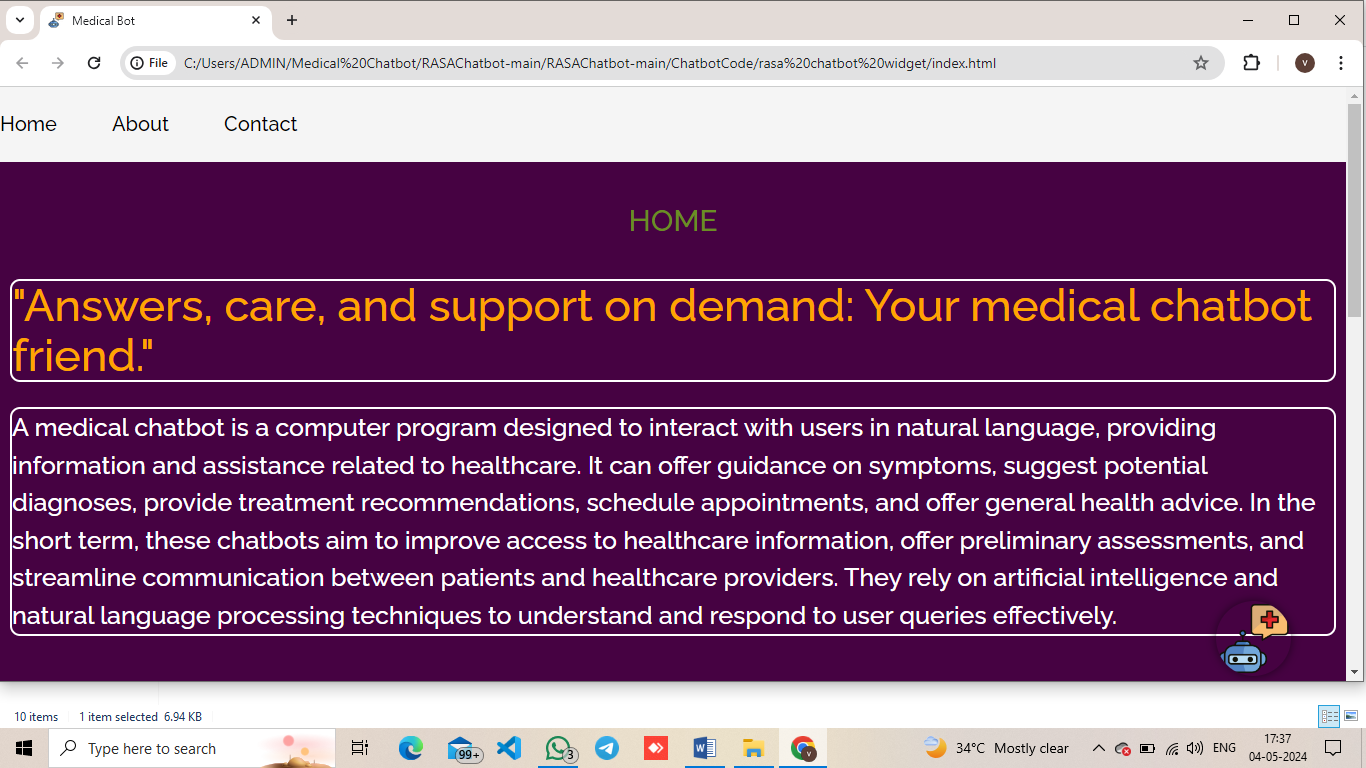
background: #b5b5b5;

border-radius: 10px;

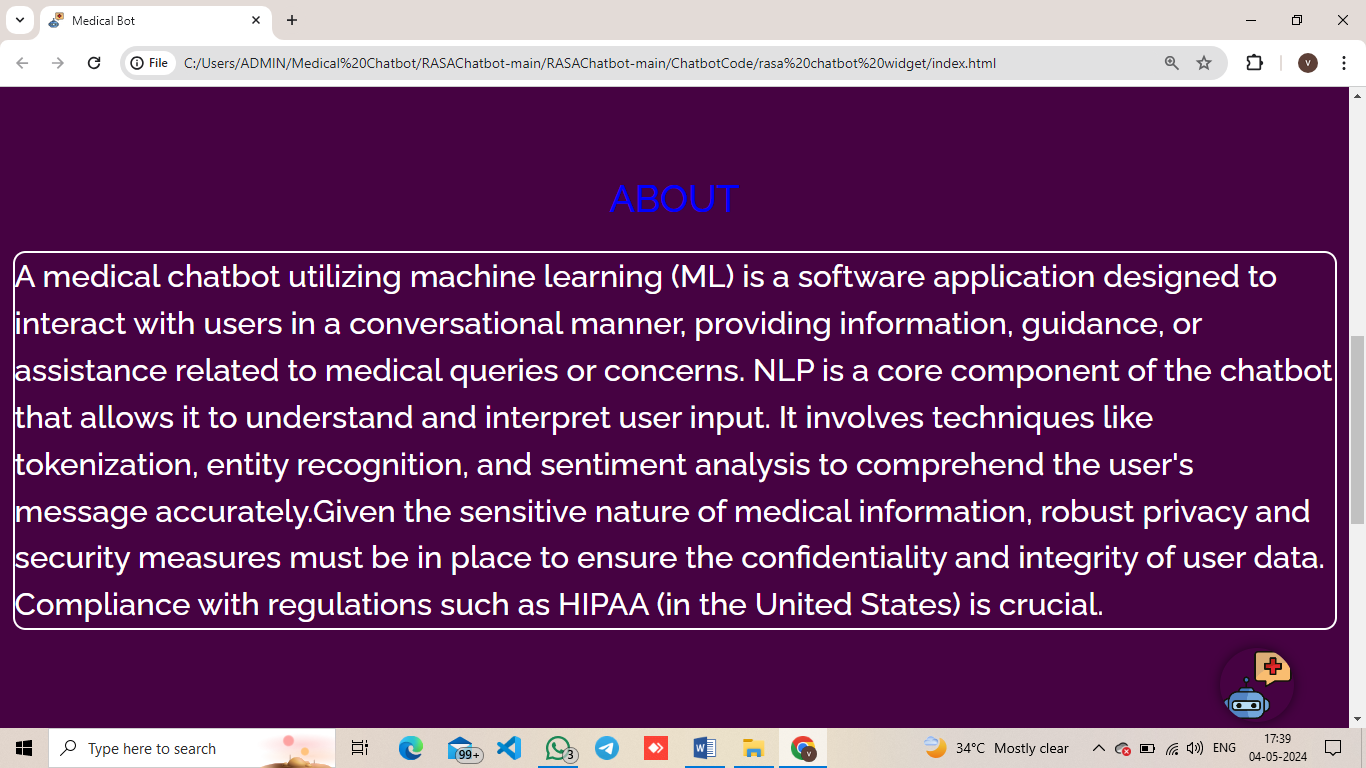
}

**APPENDIX B**

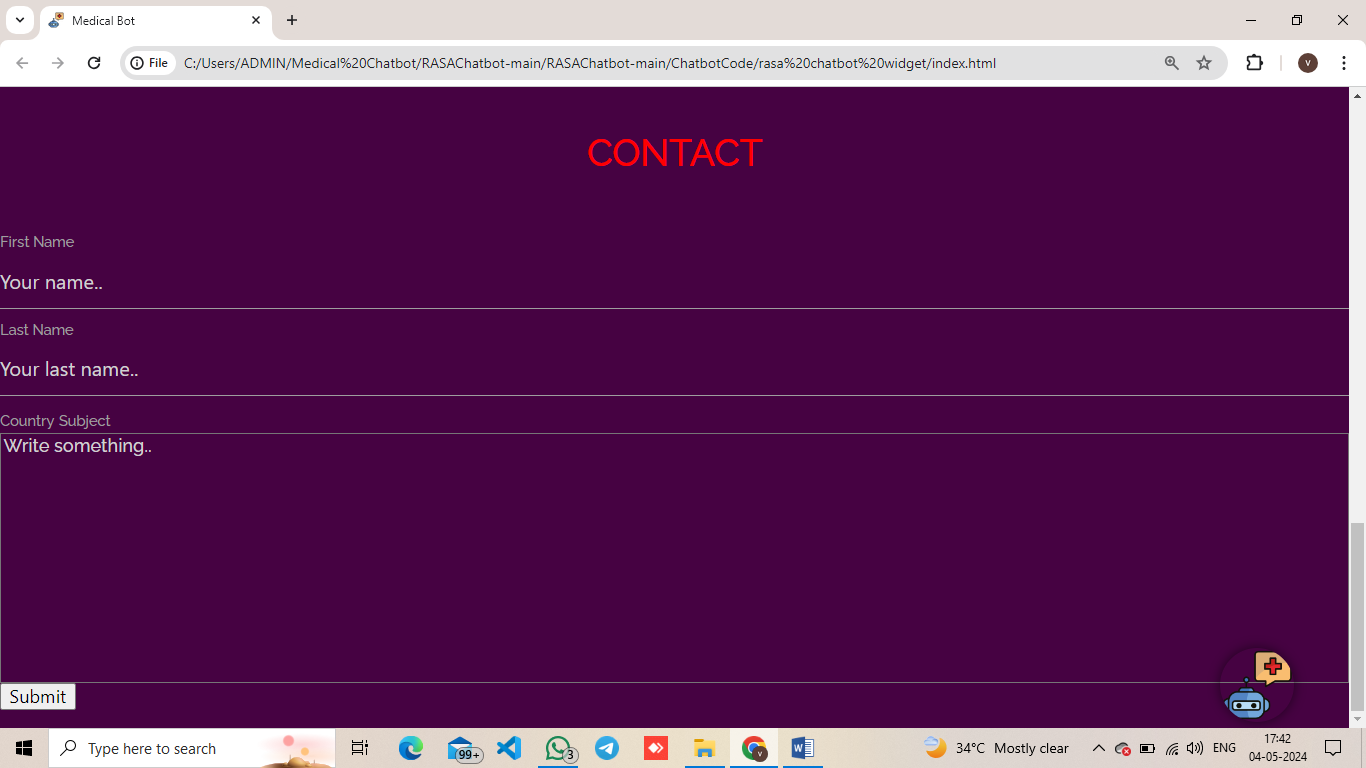
**SCREENSHOTS:**

****

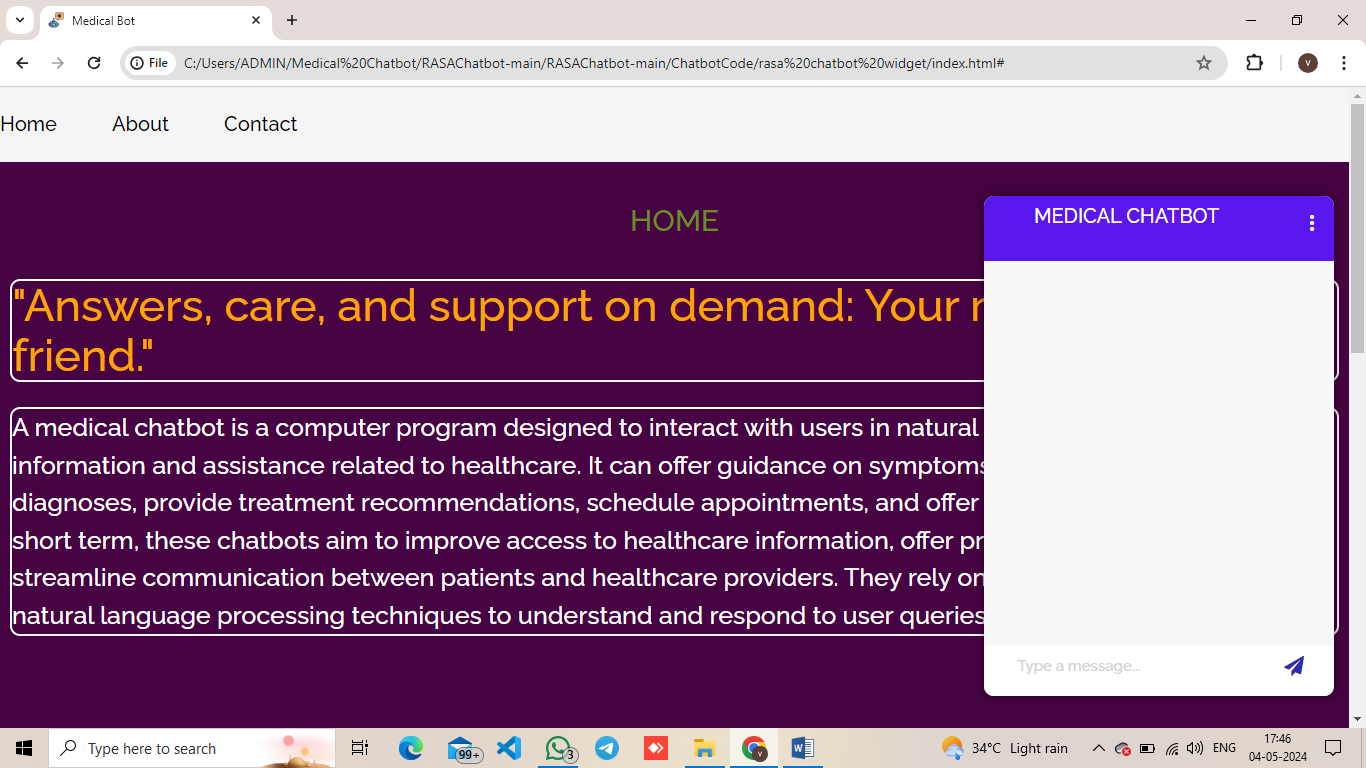
**Figure: Appendix-B-1-Home page**

****

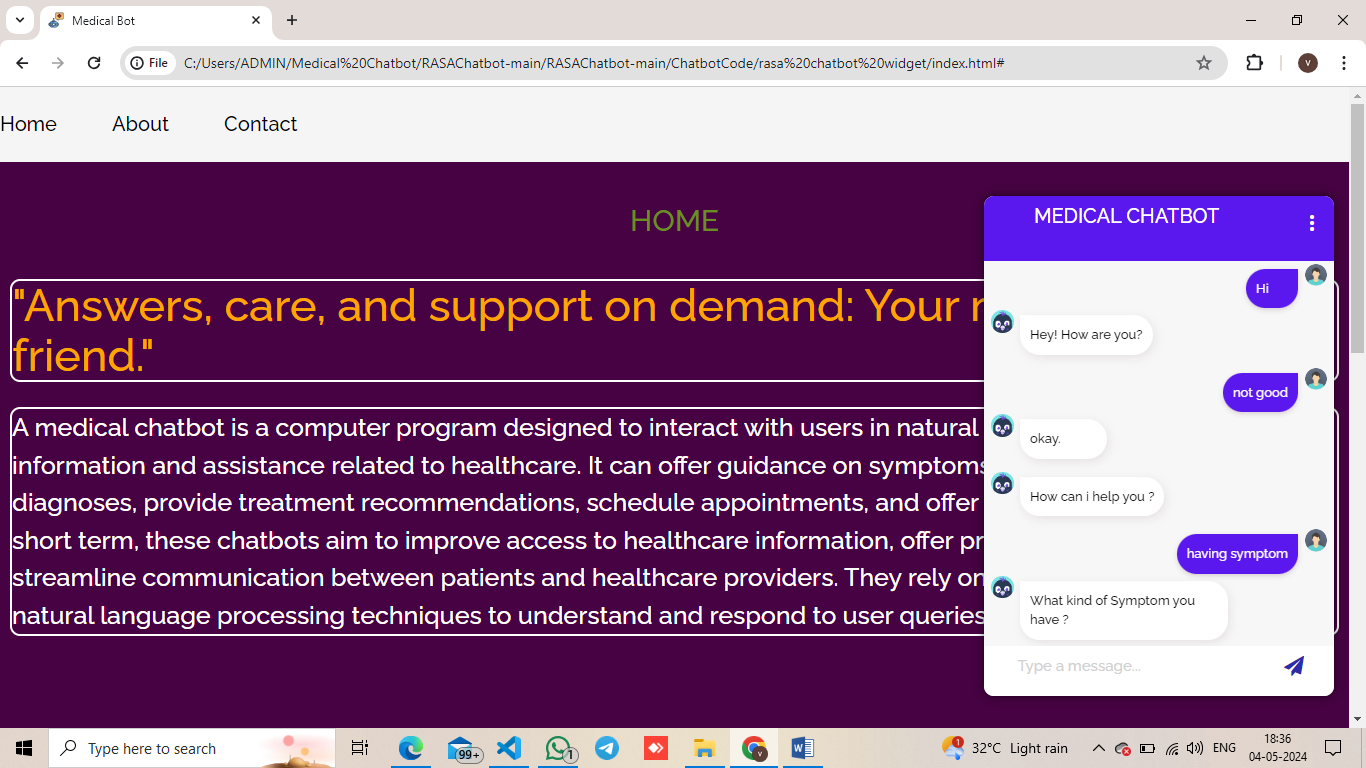
**Figure: Appendix-B-2-About page**

****

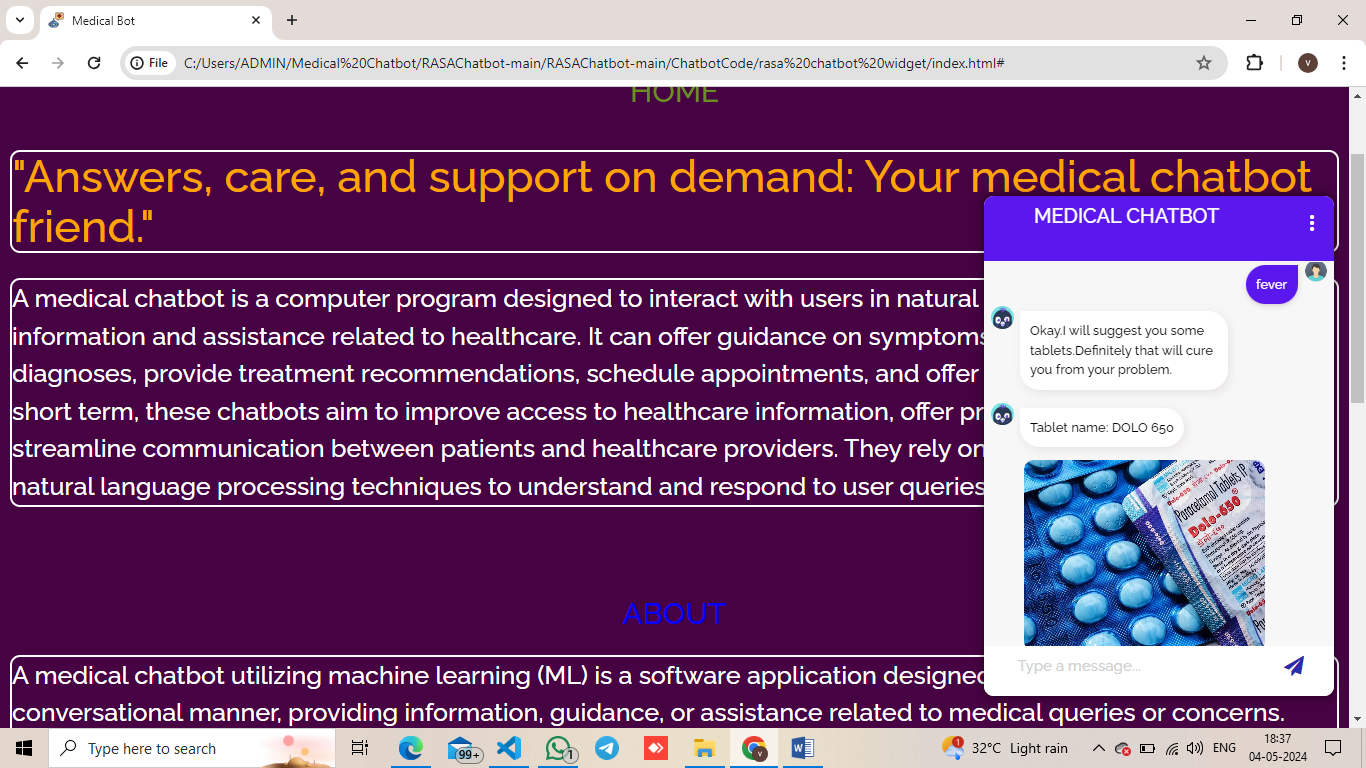
**Figure: Appendix-B-3-Contact page**

****

**Figure: Appendix-B-4-Chatbot Screen**

****

**Figure: Appendix-B-5-Chatbot output -1**

****

**Figure: Appendix-B-6-Chatbot output -2**

**REFERENCES**

[1] Takeshi Kamita, Tatsuya Ito,Atsuko Matsumoto, TsunetsuguMunakata, and Tomoo Inoue, “A Chatbot System for Mental Healthcare Based on SAT Counseling Method”. 2019.

[2] Manish Bali, Samahit Mohanty, Subarna Chatterjee, Manash Sarma, Rajesh Puravankara, “Diabot: A Predictive Medical Chatbot using Ensemble Learning” 2019.

[3] Athulya N, Jeeshna K, S J Aadithyan, U Sreelakshmi, Hairunizha Alias Nisha Rose, “Healthcare Chatbot”. 2021.

[4] Sagar Badlani, Tanvi Aditya, Meet Dave, Sheetal Chaudhari, “Multilingual Healthcare Chatbot Using Machine Learning”. 2021.

[5] Hiba Hussain, Komal Aswani, Mahima Gupta, Dr. G.T.Thampi, “Implementation of Disease Prediction Chatbot and Report Analyzer using the Concepts of NLP, Machine Learning and OCR”. 2020.

[6] Umar Jameel, Hashim Khan, Aqib Anwar, “Doctor Recommendation Chatbot: A research study”. 2021.

[7] LekhaAthota, Vinod Kumar Shukla, Nitin Pandey, Ajay Rana, “Chatbot for Healthcare System Using Artificial Intelligence”.2020.

[8] NavidaBelgaumwala, Dr. Rajashekarappa,“Chatbot: A Virtual Medical Assistant”. 2019.

[9] Simon Oermann, Kathryn L McCabe, David N Milne, Rafael A Calvo1, “Application of Synchronous Text- Based Dialogue Systems in Mental Health Interventions: Systematic Review”. 2017.

[10] Abbas SaliimiLokman, JasniMohamadZain, FakultiSistemKoputer, KejuruteraanPerisian,” Designing a Chatbot for Diabetic Patients”. 2015.

**PUBLICATIONS**

**Internation Journal of Research Publication and Reviews**

**(IJRPR)**

****

****

****

****

****